


















# **Operații pe biți. Aplicații**

*Alexandru Cohal  
Noiembrie 2013*

## *Cuprins*

 <b>Reprezentarea internă a numerelor întregi</b> .....	3
 <b>Operatori logici la nivel de bit</b> .....	4
 <b>Operatorul de negație (NOT)</b> .....	4
 <b>Operatorul de conjuncție (ȘI, AND)</b> .....	4
 <b>Operatorul de disjuncție (SAU, OR)</b> .....	5
 <b>Operatorul sau exclusiv (XOR)</b> .....	5
 <b>Operatorul de deplasare la dreapta (SHIFT RIGHT)</b> .....	5
 <b>Operatorul de deplasare la stânga (SHIFT LEFT)</b> .....	6
 <b>Operații elementare la nivel de biți</b> .....	7
 <b>Transformarea unui bit în 1</b> .....	7
 <b>Transformarea unui bit în 0</b> .....	7
 <b>Testarea valorii unui bit</b> .....	7
 <b>Testarea valorilor ultimilor biți</b> .....	8
 <b>Reprezentarea mulțimilor prin vector caracteristic implementat pe biți</b> .....	8
 <b>Probleme</b> .....	9
 <b>Legături</b> .....	9
 <b>Bibliografie</b> .....	9



## Reprezentarea internă a numerelor întregi

Reprezentarea în memorie a numerelor întregi se face în *baza de numerație 2*, printr-o secvență de cifre de **0** și **1**. Această secvență poate avea o lungime de:

- **8 biți (char, unsigned char)**
- **16 biți (short int, unsigned short int)**
- **32 biți (int, unsigned int, long, unsigned long)**
- **64 biți (long long, unsigned long long)**.

Poziția aleasă pentru reprezentarea semnului unui număr, denumită *cifră de semn*, este poziția cea mai din stânga a numărului binar. Se consideră în mod convențional cifra binară **0** pentru exprimarea semnului “+” și cifra binară **1** pentru exprimarea semnului “-”.

Forma de memorare a întregilor se face prin *Reprezentarea prin Complement față de 2* astfel:

- Reprezentarea numerelor pozitive:
  - ➡ Se reprezintă numărul în baza 2
  - ➡ Se completează pozițiile rămase libere cu 0
  - ➡ (Cifra de semn este 0).
- Reprezentarea numerelor negative:
  - ➡ Se reprezintă valoarea absolută a numărului (modulul său) în baza 2
  - ➡ Se calculează *Complementul față de 1* a reprezentării obținute (biții egali cu 1 se transformă în 0, iar cei egali cu 0 se transformă în 1)
  - ➡ Se adună 1 la valoarea obținută (adunarea se va face în baza 2)
  - ➡ (Cifra de semn va deveni automat egală cu 1 după acești pași).

**Exemple:** Considerând că folosim o variabilă de tip char (8 biți), vom obține următoarele reprezentări:

	7	6	5	4	3	2	1	0
<b>+18</b>	0	0	0	1	0	0	1	0
<b>-18</b>	1	1	1	0	1	1	1	0
<b>-101</b>	1	0	0	1	1	0	1	1



## Operatori logici la nivel de bit

Operatorii logici pe biți se aplică numai **operanzilor întregi** și au ca efect aplicarea operațiilor logice care vor fi prezentate în cele ce urmează, **bit cu bit**.

**Exemplu:** Pentru exemplificarea următorilor operatori, considerăm variabilele **a** și **b** de tip **unsigned char** cu valorile **234** respectiv **89**:

	7	6	5	4	3	2	1	0
<b>a</b>	1	1	1	0	1	0	1	0
<b>b</b>	0	1	0	1	1	0	0	1



### Operatorul de negație (NOT)

**Operator:** ~

Este singurul operator pe bit **unar** și are ca rezultat *complementarea față de 1* a operandului (biții egali cu zero vor deveni unu, iar cei egali cu unu vor deveni zero).

~	0	1
	1	0

**Exemplu:**

	7	6	5	4	3	2	1	0
<b>~a</b>	0	0	0	1	0	1	0	1
<b>~b</b>	1	0	1	0	0	1	1	0



### Operatorul de conjuncție (ȘI, AND)

**Operator:** &

Este un operator binar care returnează numărul întreg a cărui reprezentare internă se obține prin conjuncția biților situați pe aceeași poziție din reprezentarea internă a operanzilor.

Este folosit pentru a pune pe valoarea logică zero (a reseta) anumiți biți din variabila luată în considerare. Astfel, folosirea acestui operator permite scoaterea în evidență a biților care ne interesează.

&	0	1
0	0	0
1	0	1

**Exemplu:**

	7	6	5	4	3	2	1	0
<b>a &amp; b</b>	0	1	0	0	1	0	0	0



## Operatorul de disjuncție (SAU, OR)

### Operator: |

Este un operator binar care returnează numărul întreg a cărui reprezentare internă se obține prin disjuncția biților situați pe aceeași poziție din reprezentarea internă a operanzilor.

Este folosit pentru a pune pe valoarea logică unu (a seta) toți biții care au valoarea unu din cel de-al doilea operand.

	0	1
0	0	1
1	1	1

### Exemplu:

		7	6	5	4	3	2	1	0
a   b	1	1	1	1	1	0	1	1	



## Operatorul “SAU - EXCLUSIV” (XOR)

### Operator: ^

Realizează suma modulo 2 (suma aritmetică în baza 2) a operanzilor.

^	0	1
0	0	1
1	1	0

### Exemplu:

		7	6	5	4	3	2	1	0
a ^ b	0	1	0	0	1	1	0	0	



## Operatorul de deplasare la dreapta (SHIFT RIGHT)

### Operator: >>

Este un operator binar care returnează numărul întreg a cărui reprezentare este obținută din deplasarea la dreapta a reprezentării interne a primului operand cu un număr de biți egal cu al doilea operand.

Prin deplasarea la dreapta biții de pe pozițiile cele mai ne semnificative (pozițiile 0, 1,...) se pierd, iar pozițiile rămase libere se completează cu o extensie a bitului de semn dacă numărul a fost declarat ca fiind un tip de date *cu semn* (**signed**) (dacă numărul este pozitiv completarea se face cu 0, iar dacă este negativ completarea se face cu 1) sau cu zerouri dacă numărul a fost declarat ca fiind un tip de date *fără semn* (**unsigned**).

Deplasarea cu o poziție la dreapta este echivalentă cu împărțirea primului operand la doi.

### Exemplu:

		7	6	5	4	3	2	1	0
a >> 2	0	0	1	1	1	0	1	0	



## Operatorul de deplasare la stânga (SHIFT LEFT)

### **Operator:** <<

Este un operator binar care returnează numărul întreg a cărui reprezentare este obținută din deplasarea la stânga a reprezentării interne a primului operand cu un număr de biți egal cu al doilea operand.

Prin deplasarea la stânga biții de pe pozițiile cele mai semnificative se pierd, iar pozițiile rămase libere se completează cu zerouri.

Deplasarea cu o poziție la stânga este echivalentă cu *înmulțirea* primul operand cu *doi*.

### **Exemplu:**

	7	6	5	4	3	2	1	0
<b>b &lt;&lt; 4</b>	1	0	0	1	0	0	0	0



## Operații elementare la nivel de biți

Vom folosi în continuare toate variabilele ca fiind de tip **unsigned char** (8 biți, intervalul de valori posibile fiind  $[0, 255]$  ).



### Transformarea unui bit în 1

Pentru a transforma al  $k$ -lea bit al unui număr  $x$  în valoarea **1** vom aplica *operatorul de disjuncție (SAU)* dintre numărul considerat  $x$  și o *mască logică*  $m$ . Această *mască logică* va avea toți biții egali cu **0**, mai puțin al  $k$ -lea bit care va fi egal cu **1**.

Astfel, indiferent de valoarea inițială a bitului  $k$  din numărul  $x$ , după aplicarea operatorului de disjuncție dintre numărul  $x$  și masca  $m$ , valoarea bitului  $k$  din numărul  $x$  va fi egală cu **1**.

```
m = ( 1 << k );
x = ( x | m );
```



### Transformarea unui bit în 0

Pentru a transforma al  $k$ -lea bit al unui număr  $x$  în valoarea **0** vom aplica *operatorul de conjuncție (ȘI)* dintre numărul considerat  $x$  și o *mască logică*  $m$ . Această *mască logică* va avea toți biții egali cu **1**, mai puțin al  $k$ -lea bit care va fi egal cu **0**.

Astfel, indiferent de valoarea inițială a bitului  $k$  din numărul  $x$ , după aplicarea operatorului de conjuncție dintre numărul  $x$  și masca  $m$ , valoarea bitului  $k$  din numărul  $x$  va fi egală cu **0**.

```
m = 255 - ( 1 << k );
x = ( x & m );
```



### Testarea valorii unui bit

Pentru a vedea ce valoare are al  $k$ -lea bit al unui număr  $x$  vom aplica *operatorul de conjuncție (ȘI)* dintre numărul considerat  $x$  și o *mască logică*  $m$ . Această *mască logică* va avea toți biții egali cu **0**, mai puțin al  $k$ -lea bit care va fi egal cu **1**.

Astfel, după aplicarea operatorului de conjuncție dintre numărul  $x$  și masca  $m$  vom obține fie numărul **0** (caz în care bitul testat are valoarea **0**) fie valoarea  $2^k$  (caz în care bitul testat are valoarea **1**).

```
m = ( 1 << k );
val = ( x & m );
```



## Testarea valorilor ultimilor biți

Similar cu testarea valorii unui singur bit se face și testarea valorilor ultimilor  $b$  biți. *Masca logică*  $m$  va avea toți biții egali cu  $0$  mai puțin ultimii  $b$  biți care vor fi egali cu  $1$ . Valoarea acestor ultimi  $b$  biți este echivalentă cu restul împărțirii numărului  $x$  la  $2^b$ .

```
m = ( 1 << b ) - 1;  
val = ( x & m );
```



## Reprezentarea mulțimilor prin vector caracteristic implementat pe biți

O metodă de a reprezenta o submulțime  $S$  a mulțimii  $\{0, 1, 2, \dots, n\}$ , unde  $n$  este un număr natural, este *vectorul caracteristic*.

Vectorul caracteristic poate fi implementat ca un vector  $v$  cu  $n$  componente, unde  $v[x]$  are valoarea  $1$  dacă elementul  $x$  aparține submulțimii  $S$ , respectiv valoarea  $0$  în caz contrar.

O altă metodă de implementare a vectorului caracteristic, mai eficientă din punct de vedere al spațiului folosit, este asocierea fiecărui element din mulțimea considerată  $\{0, 1, 2, \dots, n\}$  a unui bit. Astfel, bitul corespunzător unui element  $x$  are valoarea  $1$  dacă elementul  $x$  aparține submulțimii  $S$ , respectiv valoarea  $0$  în caz contrar.





## **Probleme**

- Ciurul lui Eratostene reprezentat ca vector caracteristic implementat pe biți
- Rotirea la dreapta sau la stânga cu un anumit număr de biți a unui număr memorat într-o variabilă de tip unsigned int

- [Morse](#) (Campion)
- [Cod](#) (Campion)
- [Viteza](#) (Campion)
- [Lgdrum](#) (Campion)
- [Gray](#) (Campion)
  
- [Pereti](#) (Campion)
- [Aritma](#) (Campion)
- [Radio](#) (Campion)
- [Patrăte7](#) (Campion)
- [Gradina](#) (Campion)



## **Legături**

- [Operații pe biți – Articol de pe Infoarena](#)
- [Operații pe biți – Articol realizat de prof. Dan Pracsu](#)
- [Operații pe biți – Articol realizat de prof. Dana Lica](#)
- [Despre baza de numerație 2](#)



## **Bibliografie**

- Prof. Emanuela Cerchez – Operații pe biți
- Prof. Dana Lica – Operații pe biți
- Cursul de *Programare I* predat la Facultatea de Automatică și Calculatoare Iași

**Alexandru Cohal**  
[alexandru.cohal@yahoo.com](mailto:alexandru.cohal@yahoo.com)  
[alexandru.c04@gmail.com](mailto:alexandru.c04@gmail.com)  
Noiembrie 2013